Detection of Good and Bad Sensor Nodes in the Presence of Malicious Attacks and Its Application to Data Aggregation

Anes Yessembayev, Dilip Sarkar, *Senior Member, IEEE*, and Faisal Sikder, *Member, IEEE*

read
利用LOF(一种相对密度度量方法) 做 离群点outlier检测// 可能是一个思路 // 本文扩充到多维，意义不大// 但是LOF的参考文献 有意思！

***Abstract*—Most of the sensor nodes have multiple inexpensive and unreliable sensors embedded in them. For many applications, readings from multiple sensors are aggregated. However, the presence of malicious attacks adds a challenge to sensor data aggregation. Detection of those compromised and unreliable sensors and sensor nodes is important for robust data aggregation as well as their management and maintenance. In this work, we develop a method for identification of good and bad sensor nodes and apply it for secure data aggregation algorithms. We consider altered/unreliable readings as outliers and identify them using an augmented and modified version of a local outlier factor computation method. We use the outlier detection algorithm for reliable and unreliable sensor detection and use the results from this algorithm for an unreliable sensor-node identification algorithm. We show its usefulness for secure data aggregation algorithms. Extensive evaluations of the proposed algorithm show that it identifies good and bad nodes and estimates true sensor value efficiently.**

***Index Terms*—Sensor networks, collusion attacks, good and bad sensor-node detection, local outlier factor, data aggregation.**

I. Introduction

PHENOMENAL advancement of inexpensive sensors and microcontroller technologies has significantly contributed to the development of many *devices* or *things*. Tools and protocols have been developed for connecting these *things* to create the Internet of Everything (IoE) (see [1], [2], and references therein). Most IoE devices have multiple types of sensors embedded in them. For example, the 9DOF Razor IMU[1] and Sensor Hub Booster Pack[2] used in our lab have 9 and 13 sensors, respectively [3]–[5]. But inexpensive sensors are unreliable and vulnerable to simple and malicious attacks [6], [7], because they are often physically distributed in unfriendly environments. Due to this unreliability of the readings from the sensors they are deployed redundantly [8]. Readings of multiple sensors of a type are gathered and combined by an *aggregator node*. An aggregator node not only collects readings from sensors, but also eliminates or minimizes the influence of readings from unreliable and/or compromised sensors. Secure data aggregation algorithms for sensor networks aim to provide mechanisms for eliminating or resisting data distortion. These algorithms are usually run on an aggregator node [9] or a *base station*.

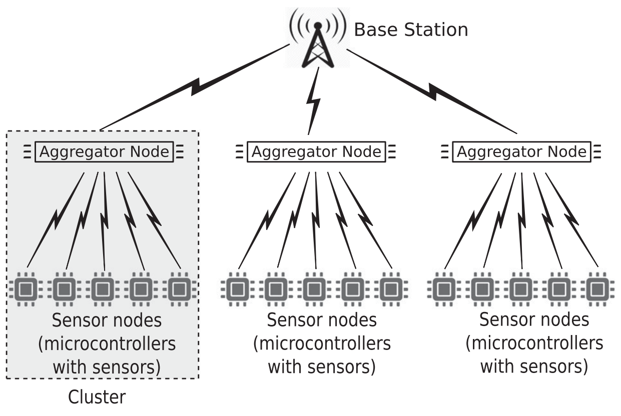

Base Station
Aggregator Node
Aggregator Node
Aggregator Node
Sensor nodes (microcontrollers with sensors)
Sensor nodes (microcontrollers with sensors)
Sensor nodes (microcontrollers with sensors)
Cluster

Fig. 1. Sensor network topology.

Any sensor readings further from the *true* value are categorized into several classes (see [10]). 1) *Noise:* data with greater variance, 2) *Spike:* data with one or more out-of-bound readings, 3) *Stuck-at:* data with quasi-zero variance, and 4) *Corrupted:* data altered by malicious attackers. In the sequel, if a sensor's reading is faraway from the *true* value of the signal, the sensor is referred to as an *unreliable* sensor; otherwise, the sensor is referred to as a *reliable* sensor.

We have developed algorithms for detection of unreliable sensors and bad nodes [11]. Also, we have used these algorithms for aggregation of data from sensors networks that may have been compromised by malicious attacker.

A. Sensor Network Model

The sensor network topology used for our work is an abstract model proposed in [12] (see Fig. 1). A sensor network consists of *a base station* and a set of *sensor clusters*. Each cluster has *an aggregator node* (or *cluster head*) that gathers data from all *sensor-nodes* connected to it. The main functions of an aggregator node are aggregation of data from its sensor nodes, computing an estimated reading, and communicating the processed data to a base station.

Manuscript received May 26, 2017; revised November 20, 2017; accepted December 21, 2017. Date of publication January 5, 2018; date of current version August 7, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Yan Lindsay Sun. *(Corresponding author: Faisal Sikder.)*

The authors are with the Department of Computer Science, University of Miami, Coral Gables, FL 33124 USA (e-mail: a.yessembayev@umiami.edu; sarkar@miami.edu; f.sikder@umiami.edu).

Digital Object Identifier 10.1109/TSIPN.2018.2790164

[1]BOOSTXL-SENSHUB BoosterPack, Texas Instruments Inc.
[2]Razor IMU-9 Degrees of Freedom, SparkFun Electronics.

Each sensor node has a micro-controller that reads output from one or more sensors. The micro-controller is equipped with a relatively small memory and has very limited computing power, while an aggregator has bigger memory and higher computing power. A base station has larger memory and computing power, in addition to communication capabilities. For our study, it is assumed that the aggregator nodes and the base station are not compromised. This assumption is justifiable, because their numbers are usually fewer and it is possible to place them in less hostile and safer locations. After receiving a batch of readings from multiple sensors, an aggregator applies some filtering technique, e.g. *Iterative Filtering*, for reducing the influences of readings from the unreliable sensors.

B. Notations Used

For concise presentation, we introduce some definitions and symbols first. Let us assume that an aggregator is associated with n sensor nodes and that each sensor node has p-types of sensors on it. A sensor node is called a *bad* node, if all sensors on the node are *unreliable*, or the micro-controller on the node is faulty, or it has been compromised by a malicious attacker who may alter readings of some or all of sensors on the node. A sensor node is called a *good* node, if all sensors on the node are *reliable* and the micro-controller on the node is faultless and it has not been compromised by a malicious attacker. Note that a sensor node could be neither bad nor good; in this situation, the microcontroller is faultless and it has not been compromised, but one or more (not all) sensors are unreliable; we refer to this type of sensor nodes as *partially-good* sensor nodes.

Let us assume that an aggregator node is assigned n sensor nodes, $\mathrm{N} = \{N_1, N_2, \cdots, N_n\}$. We also assume that sensor node has p types of sensors. This assumption implies that sensor nodes are homogeneous, which makes our presentations more manageable. For reference, let us denote the p sensors in a node Ns as $\{\mathbb{S}_{1,s}, \mathbb{S}_{2,s}, \cdots, \mathbb{S}_{p,s}\}$. A sensor $\mathbb{S}_{i,s}$ of type i at node s gathers m readings $y_{i,s}^{(t)}$ for $t \in \{1, 2, \cdots m\}$. Let $y_{i,s} = [y_{i,s}^{(1)}, y_{i,s}^{(2)}, \cdots, y_{i,s}^{(m)}]$ be a batch of m readings from sensor type i in the node N_s. A sensor node N_s sends a batch of readings $Y_s = [y_{1,s}, y_{2,s}, \cdots, y_{p,s}]$ to its aggregator node. Thus, an aggregator node receives a batch of readings $Y = [Y_1, Y_2, \cdots, Y_n]$ from n sensor nodes connected to it. Fig. 2 shows a visual interpretation of a batch of data that an aggregator node receives. The aggregator node uses these readings to solve the following problems.

C. Problem Statements

Let true and estimated values of a signal at time t for sensor type i be denoted by $r_i^{(t)}$ and $\hat{r}_i^{(t)}$, respectively.

In this work we address two problems.

1) Given a batch of readings Y from the set of sensor nodes $\{N_1, N_2, \cdots, N_n\}$, find the sensor nodes that are 'good', 'bad', and 'partially-good'—the sensor-nodes that have both reliable and unreliable sensors.
2) Given a batch of readings Y from the set of sensor nodes $\{N_1, N_2, \cdots, N_n\}$, estimate $\hat{r}_i^{(t)}$, for all $t \in \{1, 2, \cdots, m\}$ and $i \in \{1, 2, \cdots, p\}$.

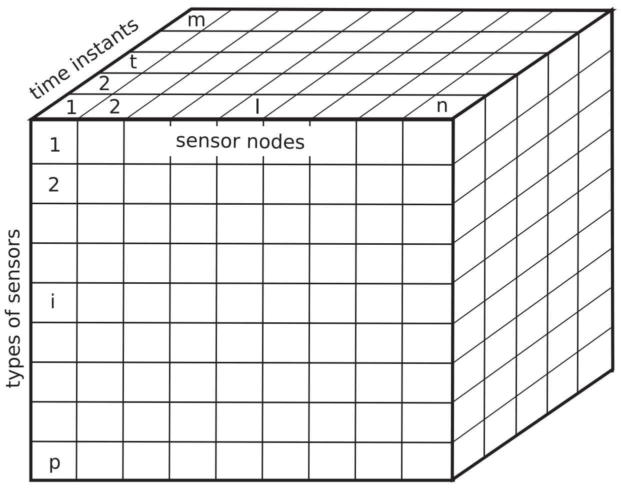


Fig. 2. A 3-D view of a batch of data to an aggregator—a vertical slice, Y_l, is readings from a sensor node and a horizontal slice, X_i, is readings from one type of sensor.

To the best of our knowledge, no attempt has been made to solve the first problem. We propose a solution to this problem in Section IV, and its performance is evaluated in Section V. Regarding the second problem, there are different approaches such as statistical estimation, iterative filtering and so on. For the work reported here we apply solution from the first problem; we eliminate readings from sensors deem to be unreliable and then use statistical estimation method described in Section II-C1.

D. Summary of Our Contributions

1) We show that if a sensor node has p sensors (one from each of the p-types) and each is designated either reliable or unreliable, then readings from n sensor-nodes (at a given time) may form as many as $\min(n, 2^p)$ clusters in p-dimension space. If $n < 2^p$, then number of clusters are limited to n, that is, readings from each sensor-node may form a cluster of its own. Thus, clustering of p-dimension data may not be used to identify reliable and unreliable sensor nodes.
2) We also analytically show that for each sensor type, if no more than n_b of the n sensors are unreliable, then the number of clusters could be very large. For instance, Example 3 in Section IV-A shows that if (i) data is received from 10 sensor-nodes, (ii) a sensor node has 4 sensors, and (iii) no more than 2 sensors of a given type can be bad, then number of clusters can be as many as 7. *Thus, clustering algorithms may not be useful for the first problem I-C.*
3) To overcome a problem that may emerge from clustering p-dimension data, we partition the readings by sensor type and then apply a variant of *Local Outlier Factor* (LOF) algorithm to classify the readings into two groups: reliable and unreliable readings. To the best of our knowledge, we are the first to apply LOF algorithm to categorize sensor-readings into reliable and unreliable classes. Once we have detected unreliable and reliable sensors, we use set union and set intersection operations to identify reliable, unreliable, and partially-reliable sensor nodes.

4) Finally, we empirically evaluate performance of our proposed method for detection of reliable and unreliable sensors as well as reliable and unreliable sensor-nodes. Moreover, we empirically show that performance of data aggregation algorithms can be significantly improved if unreliable readings are eliminated using our proposed algorithm.

The rest of the paper is organized as follows. We review sensor anomalies and data aggregation methods in Section II. Next, present a local outlier factor computation algorithm and three LOF-based outlier detection algorithms in Section III. A method for identification of the good and the bad sensor nodes is proposed in Section IV. There we also introduce necessary formalism for the proposed method. We evaluate performances of the proposed algorithms for identification of the good and the bad sensor-nodes in Section V. In Section VI, we propose several two-phase data aggregation algorithms — the first phase of all these algorithms applies reliable and unreliable sensor detection algorithms presented in Section IV-C. We also empirically evaluate performances of these algorithms. Section VII provides concluding remarks and describes potential future work.

II. Review of Sensor Anomalies and Data Aggregation Methods

A. *Review of Sensor Anomalies*

Effects of sensor data aggregation because of possible anomalies introduced in Section I are reviewed now. For a fast visual comparison, Fig. 3 shows an overview of sensor anomalies and their known solutions. For ease of comparison three columns are used—the left-most and right-most columns are for the sensors and the aggregator nodes, respectively; the middle column is for the attributes in the rows. It illustrates problems they encounter, how those problems are dealt with (solutions), and expected results. Several symbols are introduces for ease of visualization. The symbols are annotated at the bottom row of the diagram.

In general, low cost sensors are unreliable and data aggregator nodes have small memory, low computation capacity, limited power, and communication bandwidth. The major anomalies or problems with sensors are stuck-at fault ([10], [13], [14]), noise and spikes ([15]–[19]), and attacks—simple or malicious ([9], [20], [21]). These anomalies present a data aggregator node varied level of challenges.

If the readings from the sensors have only random noise, a simple statistical estimation with or without elimination of outliers is expected to yield a good estimation of the true signal value (see Section II-C1 for a brief description). On the other extreme is coping with malicious attacks, where sensors and/or sensor nodes are compromised by adversaries. Iterative filtering algorithms are quite effective for reducing effect of simple attack. We briefly review this method in Section II-C2. A malicious attacker with the knowledge of the filtering method used by the aggregator circumvent elimination of his attack by altering readings shrewdly [9]. To overcome effect of malicious attack, a robust aggregation method was proposed and evaluated in [9]. The basic principle of the method is briefly reviewed in Section II-C3.

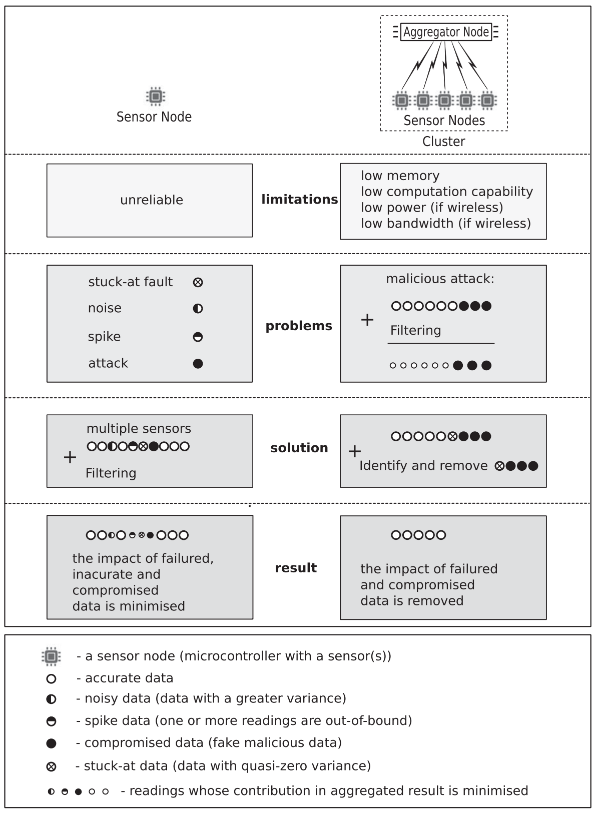

Sensor Node
Aggregator Node
Sensor Nodes
Cluster
unreliable
limitations
low memory
low computation capability
low power (if wireless)
low bandwidth (if wireless)
stuck-at fault ⊗
noise ◐
spike
attack ●
problems
malicious attack:
○○○○○○●●●
+
Filtering
○○○○○○●●●
multiple sensors
+
Filtering
solution
○○○○○⊗●●●
+
Identify and remove ⊗●●●
result
the impact of failured, inacurate and compromised data is minimised
○○○○○
the impact of failured and compromised data is removed
- a sensor node (microcontroller with a sensor(s))
○ - accurate data
◐ - noisy data (data with a greater variance)
- spike data (one or more readings are out-of-bound)
● - compromised data (fake malicious data)
⊗ - stuck-at data (data with quasi-zero variance)
- readings whose contribution in aggregated result is minimised

Fig. 3. Sensor anomalies and methods for elimination or reduction of their effects.

B. *Review of Faulty Sensor Detection Methods*

Based on algorithms, faulty sensor detection methods have two primary categories [8]: (i) centralized detection methods, and (ii) distributed detection methods. In the centralized approaches, a sink node or aggregator node collects data and analyzes the data to find the faulty sensors [22]–[24]. In the distributed approaches, each sensor node analyzes data and creates a regional-view of the faulty sensors. It then transmits this information to neighbors; thus, state of all sensor nodes propagates to all sensor nodes in the network [25]–[29] with a delay. Maximum propagation delay depends on the diameter of the network.

Distributed fault-detection algorithms in [16], [25], [28] are iterative algorithms, because each node iteratively detects faulty sensors and transmits that information to other nodes iteratively. The outlier detection method in [30] uses Bayesian approach. Another distributed approach for detection of failed sensor is test-based, where each node is assigned some specific task to determine states of the sensors [25], [29], [31].

The main differences between our proposed method and existing studies are: (i) we assume a node has multiple sensors, and (ii) we use LOF based algorithm for unreliable sensor identification.

C. *Review of Data Aggregation Methods*

1) Statistical Estimation: If the readings from the sensors have no outliers, a simple statistical method for estimating true

TABLE I
ITERATIVE FILTERING ALGORITHM CONVERGES TO THE COLLUDED READING PROVIDED BY SENSOR $\mathbb{S}_{10}$

	Readings from Sensor $\mathbb{S}_i$										aggregate values
	$\mathbb{S}_1$	$\mathbb{S}_2$	$\mathbb{S}_3$	$\mathbb{S}_4$	$\mathbb{S}_5$	$\mathbb{S}_6$	$\mathbb{S}_7$	$\mathbb{S}_8$	$\mathbb{S}_9$	$\mathbb{S}_{10}$	
	20.2	19.9	20.1	20.0	19.7	20.2	20.4	39	41	**24.6**	
Iteration	Sensor weights										
1	1	1	1	1	1	1	1	1	1	1	24.510
2	0.054	0.047	0.051	0.049	0.043	0.054	0.059	0.005	0.004	123.457	24.588
3	0.052	0.046	0.050	0.048	0.042	0.052	0.057	0.005	0.004	6962.460	**24.600**

values is quite effective. The main assumption for the method is that every sensor has a certain error in its readings, but the readings of a sensor that has lower temporal variation is given higher weight. Thus, the inverse of the variance v_s of readings of a sensor s is used to determine the weight w_s that is applied to the readings of the sensor for estimating true signal values, that is, $w_s = \frac{1}{v_s} / \sum_{i=1}^{n} \frac{1}{v_i}$. Therefore, estimates of the true values are obtained using the equation below.

方差的导数 作为权重
s维度 和 i维度
越平稳 越有用

$$\hat{r}^{(t)} = \sum_{s=1}^{n} w_s \cdot x_s^{(t)} \tag{1}$$

This method should be used only if the sensor readings are free of outliers or outliers have been identified and removed from the readings.

用在网络安全里？

2) Iterative Filtering Algorithms: The most studied method related to our data aggregation work is *Iterative Filtering* (IF) algorithms (see [32] and [33], and references therein), where an initial weight is assigned to each reading for estimating true signal value. In the next iteration, new weights are calculated for each sensor and they are used to compute the next estimate. The process continues until the difference between two consecutive estimates is below a predetermined threshold.

affine
仿射函数

The IF algorithm introduced in [33] converges if inverse, exponential, and affine functions are used as discriminant. Six iterative algorithms introduced in [34] for calculating ranking using reputation of users. Ayday *et al.* developed an iterative algorithm with probabilistic and belief propagation-based approach (see details in [32]). However, a basic limitation of all IF algorithms is that they are primarily aimed against simple attacks and do not consider severe malicious attacks [9]. Let us consider an example to illustrate this issue.

Example 1: Let us consider a dataset with $m = 1$ readings from $n = 10$ sensor nodes, and one type of sensors, i.e., $p = 1$. Sensors $\mathbb{S}_1$–$\mathbb{S}_7$ are providing their *true* readings to the aggregator, while $\mathbb{S}_8$–$\mathbb{S}_{10}$ are under the influence of a malicious attacker and providing *manipulated* readings. The reading from $\mathbb{S}_{10}$ is close to the average of all readings of all the sensors. Table I shows how the algorithm converges quickly to the reading from $\mathbb{S}_{10}$. This happens because the algorithm assigns equal initial weights for each sensor at the first round.

3) Enhanced Iterative Filtering Algorithm [9]: To reduce influence of collusion attack, Rezvani *et al.* proposed a *Robust Data Aggregation Method* (RDAM), which is an improvement to IF method. In this method, unequal initial weights are calculated using the readings available to the aggregator. This enhancement not only makes an IF algorithm collusion attack resistant, but it also reduces the number of required iterations to converge to an estimated value. The method is based on the assumption that the distribution of stochastic components of sensor errors is known or can be estimated.

Next, after introducing the concept of local outlier factor we present algorithms for outlier detection. First, an outline of an algorithm for calculation of local outlier factor is presented and then, we present our outlier detection algorithms.

III. LOCAL OUTLIER DETECTION ALGORITHMS

Detection of outliers in a dataset is a well-studied subject. Before we provide the algorithm, next a very brief overview of the existing outlier detection methods is provided.

In general, outlier detection methods form five main classes: statistical, nearest neighbor, clustering, classification, and spectral decomposition [18]. In statistical methods outliers are defined as objects that do not fit in assumed distribution [17], [19], [35]. Nearest neighbor methods use a distance as a mean to distinguish outliers [16],[36]. Clustering algorithms use similarity metrics [37]. Most popular and probably old clustering algorithm is *k-means.*

However, the information age, especially big-data and *Internet of Everything* (IoE), has seen renewed interest in the subject to fit the current needs. The newest and popular for detecting outliers in *big data* is *local outlier method* [38]. In our study, we use a variant of a recently proposed method for local outlier method [36].

A. Introduction to Local Outlier Factor Concepts

The *Local Outlier Factor* algorithm proposed by Breunig M. *et al.* for finding anomalous objects [36] measures the local deviation of a given object $o_i \in O$ with respect to its neighbors in O [36]. The idea of LOF is based on the concept of local density. The algorithm requires the user to provide only one parameter k, the minimum number of local neighbors of an object to be considered for computing LOF. For $k = 5$, Fig. 4 illustrates that the point o_9, for example, has a high LOF; because to compute its LOF five objects $\{o_5, o_6, o_7, o_8, o_{10}\}$ must be considered and the objects o_5, o_6, and o_7 are far from o_9. As shown in the figure, LOF for all the points o_i for $i \in \{1, 2, \cdots 10\}$ are 1.41, 1.23, 0.87, 0.67, 0.87, 1.23, 1.41, 1.93, 2.36, and 2.57, respectively.

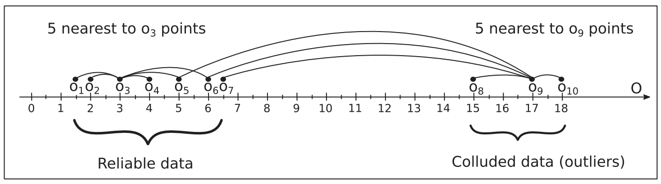


Fig. 4. Illustration of the concept of local outlier factor.

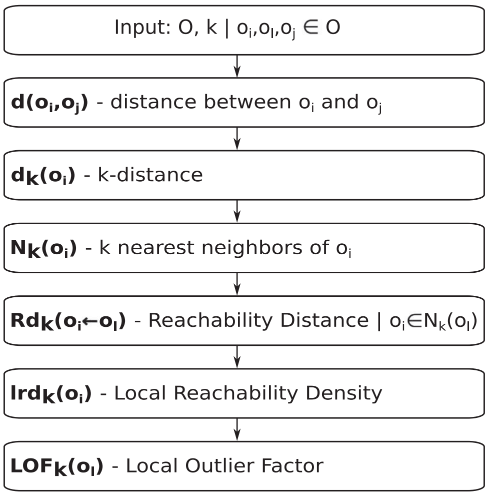

Input: O, k | oi,ol,oj ∈ O
d(oi,oj) - distance between oi and oj
dk(oi) - k-distance
Nk(oi) - k nearest neighbors of oi
Rdk(oi←ol) - Reachability Distance | oi∈Nk(ol)
lrdk(oi) - Local Reachability Density
LOFk(ol) - Local Outlier Factor

Fig. 5. Steps for computing local outlier factor.

有误！！

Fig. 5 shows six steps for computation of LOF. For conserving space we do not provide details of the algorithm. Interested reader can find the algorithm in [36]. Here we provide definitions of the terms shown in the steps of the algorithm. This algorithm is suitable for our problem, because it can be implemented using $O(n \log n)$ computation steps.

Let us start with an informal definition of an outlier.

Definition 1. (Informal definition [39]): An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism. ■

For the presentation in the rest of this section the dataset is denoted with $O = \{o_1, o_2, \cdots, o_n\}$. Next, we provide six definitions, one corresponding to each step of the LOF algorithm shown in Fig. 5.

Definition 2. (Distance $d(o_i, o_j)$): Distance between two points o_i and o_j, denoted as $d(o_i, o_j)$, is a nonnegative number, such that (i) $d(o_i, o_j) = 0$, (ii) $d(o_i, o_j) > 0$, if $o_i \neq o_j$, and (iii) for three distinct points o_i, o_j, and o_l if $d(o_i, o_j) = x$ and $d(o_j, o_l) = y$, then $d(o_i, o_l) \leq (x + y)$. ■

Above definition encompasses many methods for measuring the distance between two points, including Euclidean distance. *In general, computation time for all-pair distances is $O(n^2)$ for n points. But for one-dimension data LOF computation has $O(n \log n)$ implementation [20].*

The next step of LOF computation is to find *k*-distance, $d_k(o_i)$. It should be noted that $d_k(o_i)$ is neither distance of the furthest point nor the number of points in the neighborhood of the point o_i. It is true that the neighborhood of o_i will have at least k points, but the actual number may be much more than k.

Definition 3. (k-distance $d_k(o_i)$ [36]): For any positive integer k, the k-distance of object o_i, denoted as $d_k(o_j)$, is defined as the distance $d(o_i, o_l)$ between o_i and an object $o_l \in O$ such that: (i) for at least k objects $o_j \in O \setminus \{o_i\}$ it holds that $d(o_i, o_j) \leq d(o_i, o_l)$, and (ii) for at most $(k-1)$ objects $q \in O \setminus \{o_i\}$ it holds that $d(o_i, o_j) < d(o_i, o_l)$. ■

After $d_k(\cdot)$, the k-distance, calculation of all the points, the next step is to compute $N_k(\cdot)$. Intuitively, $N_k(o_i)$ is the set of all points in the neighborhood of o_i whose distance is less than or equal to $d_k(o_i)$. Formally, $N_k(o_i)$ is defined as.

Definition 4. (k-distance neighbors $N_k(o_i)$ [36]*):* Given the k-distance $d_k(o_i)$ of o_i, k-distance neighbors of an object o_i are the objects whose distance from o_i is not greater than the k-distance of o_i, that is, $N_k(o_i) = \{o_j \in O \setminus \{o_i\} \mid d(o_i, o_j) \leq d_k(o_i)\}$. ■

Definition 5. (Reachability Distance Rd_k*):* The Reachability Distance of object o_i with respect to another object o_l is defined as

$$Rd_k(o_i \leftarrow o_l) = \max\{d_k(o_l), d(o_i, o_l), \epsilon\}, \tag{2}$$

where ϵ is a small constant that is introduced to avoid division by zero operation in further calculations. ■

The above definition extends the definition in [36] to accommodate the case when $Rd_k(o_i \leftarrow o_l)$ may be zero, which may result in division by 'zero'.

Definition 6. (Local Reachability Density lrd_k *[36]):* The *reachability density* of an object o_i is defined as

$$lrd_k(o_i) = \frac{|N_k(o_i)|}{\sum_{o_j \in N_k(o_i)} Rd_k(o_j \leftarrow o_i)}, \tag{3}$$

where $|N_k(o_i)|$ is the number of objects in $N_k(o_i)$. ■

Definition 7. (Local Outlier Factor $LOF_k(o_i)$ [36]): The *local outlier factor* of o_i is defined as

$$LOF_k(o_i) = \frac{\sum_{o_j \in N_k(o_i)} \frac{lrd_k(o_j)}{lrd_k(o_i)}}{|N_k(o_i)|} = \left(\frac{\sum_{o_j \in N_k(o_i)} lrd_k(o_j)}{|N_k(o_i)| * lrd_k(o_i)}\right). \tag{4}$$

类似于一个相对密度？
相对密度越低 越正常 反之则越离群outlier

■

1) Selection of k in the Presence of Collusion Attack: As discussed in the original work, selection of optimal value is important. The optimal value of k is defined as $k = (n_g - 1)$, where n_g is the number of objects that are not outliers. For our case the simplest way to select k is based on the assumption: *there are considerably more 'reliable' observations than unreliable or 'compromised' observations (outliers/anomalies) in the data.* We use this hypothesis and set $k = \lfloor (n-1)/2 \rfloor$.

B. Detection of Outliers

In this section, it is assumed that a batch of data Y received at an aggregator node has been partitioned by type of sensors (see Section IV-B for partitioning procedure). The dataset for each type of sensors is handled separately for identification of outliers in the dataset. Thus, for simplifying notations we use only index for the sensor nodes and omit index for sensor type. For example, a reading $x_{i,s}^{(t)}$ from sensor type i from node N_s at time t is denoted as $x_s^{(t)}$.

Algorithm 1: Elimination of Outliers in a Single Step.

```
1: procedure OutliersDetectSingleStep
2:     Input: $X[m, n]$; Output: $S'$
3:     Compute $k = \lfloor (n-1)/2 \rfloor$
4:     Compute $\overline{x}_s = \frac{\sum_{t=1}^{m} x_s^{(t)}}{m}$ for each sensor $1 \leq s \leq n$
5:     Compute $LOF_k(\overline{x}_s)$ for each sensor $1 \leq s \leq n$
6:     Compute $\overline{LOF}_k(\overline{X}) = \frac{\sum_{s=1}^{n} LOF_k(\overline{x}_s)}{n}$
7:     for $s = 1 : n$ do
8:     if $LOF(\overline{x}_s) < \overline{LOF}_k(\overline{X})$ $S' = S' + \{s\}$
9: end procedure
```

认可小的LOF

The first two algorithms (DOSS, DOI) described next are for detection of anomalous sensors and hence use average of readings from a sensor. The third algorithm (DORV) detects outliers for a given time instance.

1) Detection of Outliers in a Single Step (DOSS): As the name suggests the proposed algorithm detects outliers (that is, data from suspected anomalous sensors) in a single step. The outliers are excluded for estimation of true signal values. First, a sequence of $LOF_k(\overline{x}_1), LOF_k(\overline{x}_2), ..., LOF_k(\overline{x}_n)$ is computed for a vector of averages $\bar{X} = \overline{x}_1, \overline{x}_2, ..., \overline{x}_n$. As discussed earlier, higher the value of LOF of an object, higher the probability of the object being from an anomalous sensor. The main challenge here is to set a criteria to split the sequence of LOF values so that only readings from anomalous sensors are excluded.

Our extensive empirical observations suggest that LOF_k of an outlier is above $\overline{LOF}_k$, which is defined as the average of all LOF_k values:

$$\overline{LOF}_k(\bar{X}) = \frac{1}{n}\sum_{s=1}^{n} LOF_k(\overline{x}_s) \tag{5}$$

A pseudo code description of the algorithm is shown in Algorithm 1. Input to the algorithm is the dataset X with m observations for each of n sensors and output is a vector S' with indices of sensor nodes deemed to be reliable. First, the algorithm calculates $k = \lfloor (n-1)/2 \rfloor$. Then the algorithm computes $\overline{x}_s$, an average of all m readings for a given sensor-type at node N_s. This is repeated for all n sensor nodes. Now we have a vector $\overline{X}[n]$ of n data points. Third, compute $LOF_k(\overline{x}_s)$ for each element of $\overline{X}$. Next, calculate the average $\overline{LOF}_k(\overline{X})$ of all $LOF_k(\overline{x}_s)$. Then, compare each $LOF_k(\overline{x}_s)$ value with the $\overline{LOF}_k(\overline{X})$. Those sensors whose $LOF_k(\overline{x}_s)$ values are below the average are assumed to be reliable.

a) Time complexity: All `Steps`, except `Step 5`, of Algorithm 1 have constant or n time complexity. The time complexity of the `Step 5` is $O(n \log n)$ [20]. Thus, overall time complexity of the algorithm is $O(n \log n)$.

2) Detection of Outliers Iteratively (DOI): This algorithm is a variation of the previous algorithm, where one outlier with the maximum LOF value is removed at each iteration. At the beginning of each iteration, LOF values of the remaining data set are calculated and the one data point with highest LOF is removed. Since the algorithm computes outliers repeatedly, it has a higher time complexity, but in some cases identification of outliers is more accurate. We omit the description of the algorithm since it is very similar to the previous algorithm. Interested reader can find the algorithm and its performance from [20].

移除大的LOF

a) Time complexity: This algorithm utilizes Algorithm 1 for detecting one outlier. Thus, time complexity of the algorithm is $O(\imath n \log n)$ for detecting $\imath$ outliers. In the worst case, $\imath$ can be $n/2$.

3) Detection of Outliers Using Row-Wise Votes (DORV): Two outlier detection algorithms presented above, compute an average of all m readings from a sensor in a node. The process is repeated n times, once for each sensor node. Then, outlier detection algorithm identifies outliers from the averages. The algorithm presented in here identifies outliers from the n readings that are reported at each time instance t.

A pseudo code description of the algorithm for Row-Wise Votes is shown in Algorithm 2. It computes $LOF_k(x_s^{(t)})$ for every observation $x_s^{(t)}$ from the sensor in the sensor node N_s. The average $\overline{LOF}(x^{(t)})$ value is calculated for every vector of sensor readings $x^{(t)}$. For those readings in $x^{(t)}$ that have the LOF values ~~below the threshold~~ are deemed to be reliable, and hence for them votes of '1' are assigned to the corresponding elements of vote array, $votes[m, n]$; other elements of $votes[m, n]$ are assigned votes of '0'. This creates a table of votes (see example in the Table III).

The average value of all votes for a sensor-type at a sensor node is referred to *reliability*. Sensors that have *reliability* greater than a certain threshold are considered as reliable. Output of the algorithm is a vector S' with indices of the sensor nodes that deemed to have reliable sensors (of the type under consideration).

TABLE II
NOTATIONS USED IN THE ALGORITHMS

$e_s^{(t)}$	noise at time t for sensor at node N_s
S	a set of all sensor nodes
S'	a set of reliable sensor nodes
$\overline{X}$	average readings of all sensor nodes
$\overline{LOF}_k(X)$	an average of all $LOF_k(x_s)$
$\overline{LOF}_k(\overline{X})$	an average of all $LOF_k(\overline{x}_s)$

TABLE III
VOTES TABLE

	Sensor nodes for a given type of sensor							
instant	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
$t = 1$	1	1	0	1	1	0	1	1
$t = 2$	0	1	1	0	0	0	1	1
$t = 3$	1	1	0	0	0	1	0	0
$t = 4$	1	1	0	1	1	0	1	1
$t = 5$	1	1	0	1	1	0	1	1
$t = 6$	1	1	1	1	0	1	1	0
$t = 7$	1	0	0	0	1	0	1	1
$t = 8$	1	1	0	0	1	0	1	1
$t = 9$	1	1	1	0	0	0	1	1
$t = 10$	1	1	0	0	1	1	0	1
reliability	0.9	0.9	0.3	0.4	0.6	0.3	0.8	0.8

Algorithm 2: Outlier Elimination Using Row-Wise Votes.

```
1:  procedure OutliersDetectVotes
2:    Input: $X[m,n], threshold$; Output: $S'$
3:    for $t = 1:m$ do
4:      for $s = 1:n$ do Compute $LOF_k(x_s^{(t)})$
5:      Compute $\overline{LOF}_k(x^{(t)}) = \frac{\sum_{s=1}^{n} LOF_k(x_s^{(t)})}{n}$
6:      for $s = 1:n$ do
7:        if $LOF_k(x_s^{(t)}) < \overline{LOF}_k(x^{(t)})$ $votes_s^{(t)} = 1$
8:        else $votes_s^{(t)} = 0$
9:      for $s = 1:n$ do
10:       if $\frac{\sum_{t=1}^{m} votes_s^{(t)}}{m} > threshold$ $S' = S' + \{s\}$
11: end procedure
```

The advantage of this method is that it allows to set a threshold for excluding outliers; a higher threshold eliminates readings from a larger number of sensors, but the readings from remaining sensors are expected to have high reliability. The data shown in the Table III identify sensors in the nodes N_1, N_2, N_5, N_7, and N_8 as reliable, if threshold is set to 0.5. However, for same data only sensors in the nodes N_1, N_2, N_7, and N_8 are found to be reliable, if the threshold is set to 0.7.

a) Time complexity: The time complexity of `Steps 4 to 8` is similar to Algorithm 1, and has time complexity of $O(n \log n)$. Since these steps are repeated m times, the time complexity of `Steps 3 to 8` is $O(mn \log n)$. Because time complexity of the `Steps 9 to 10` is $O(n)$, the overall time complexity is $O(mn \log n)$.

Any of the three algorithms described above can be used for detecting anomalous sensors. Once they have been identified and labeled reliable and unreliable, Lemmas 3 and 4 in Section IV-C are applied to identify good nodes and bad nodes. Nodes that are not in the either group are partially-good/-bad nodes. Recall that these partially bad nodes have some reliable sensor(s) and some unreliable sensor(s).

Next, we present a proposed method for identification of good and bad nodes.

IV. Identifying Good and Bad Sensor Nodes

A. Clustering Sensor-Node Readings

Let us treat Y as a set of nm data-points of dimension p. A data-point from a sensor node s at time t is given by $Y_s^{(t)} = (y_{1,s}^{(t)}, y_{2,s}^{(t)} \cdots, y_{p,s}^{(t)})$. For a given time instant t the readings from n reliable and unreliable sensors naturally form several groups: 1) all sensor nodes with all reliable sensors form one group, 2) all sensor nodes with all unreliable sensors form another group, and 3) other sensor nodes form potentially several clusters as discussed next.

Let us consider an example next.

Example 2: For simplicity, in this example we assume that data are sent for only one time instant and hence the index for time is omitted. Also, it is assumed that each sensor node has two types of sensors: a temperature sensor and a dewpoint sensor. An aggregator receives 10 data points from 10 of these sensor nodes. For a given time instant, let us assume that true values of temperature and dewpoint are $70.7°F$ and $40.5°F$. As shown in Table IV, temperature readings from sensors in the nodes N_1, N_3, N_4, N_5, N_6, and N_8 are very close to the true value, $70.7°F$. Similarly, dewpoint readings from sensors in the nodes N_1, N_2, N_3, N_5, N_9, and N_{10} are very close to the true value, $40.5°F$. Thus, both sensors in the nodes N_1, N_3, and N_5 are reporting reliable readings and by our definitions, they are good nodes. Similarly, both sensors in the node N_7 are reporting unreliable readings and it is a bad node. Other nodes can be divided into two groups, one group has only reliable temperature sensors and the other group has only reliable dewpoint sensors.

TABLE IV
Readings From Temperature and Dewpoint Sensors From Ten Sensor Nodes

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
temp.	70.7	77.4	70.9	70.2	70.4	71.0	77.4	70.6	76.9	77.1
dewpoint	40.7	40.6	40.3	45.2	40.4	44.9	45.1	44.6	40.1	40.2

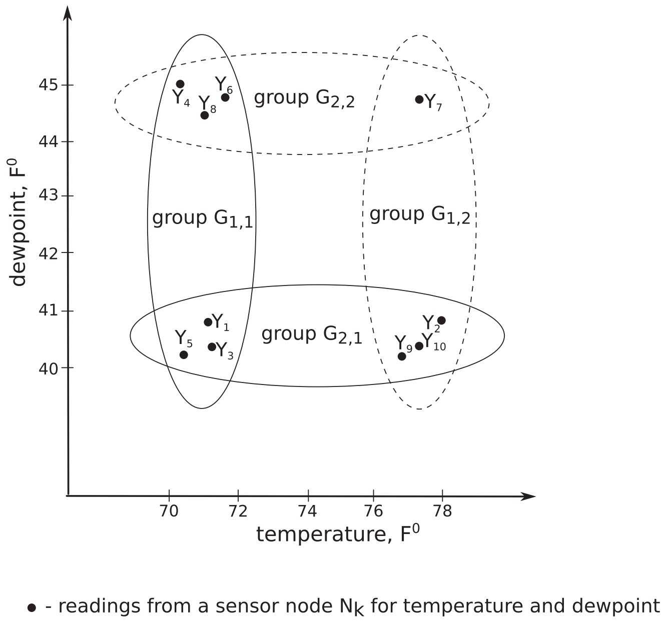


Fig. 6. Two-dimensional readings from sensor nodes.

If we treat the readings as 2-dimension data, and the data form four groups as shown in Fig. 6. Each group has data from fewer than half of the sensor nodes. But as noted earlier, for each sensor-type we have more than half of the readings from reliable sensors. *Thus, a good filtering algorithm is expected to estimate values close to true values from these readings.*

The Example 2 has shown that two types of sensors with only two classes of readings from each sensor type can form four clusters. Below we formalize the results for p-types of sensors.

Lemma 1: At time instant t, let $Y^{(t)} = \{Y_1^{(t)}, Y_2^{(t)}, \cdots, Y_n^{(t)}\}$ be a set of n data-points of dimension p from a set of n sensor nodes $\{N_1, N_2, \cdots, N_n\}$. Also, assume that for each sensor-type, sensors are grouped into two groups: reliable and unreliable. If these p-dimension n data-points are clustered,

1) The minimum number of clusters is at least 1.
2) The maximum number of clusters is at most $\min(n, 2^p)$.
3) At least one cluster has $\lceil n/2^p \rceil$ data-points. ∎

Proof: Let us consider a data-point $Y_s^{(t)} = (y_{1,s}^{(t)}, y_{2,s}^{(t)} \cdots, y_{p,s}^{(t)})$. Now, each sensor reading $y_{i,s}$ can take one of the two values, reliable or unreliable. Let us encode reading of a reliable and unreliable sensor with a 1 and 0, respectively. Thus, encoded values of readings from a node is a binary string of length p.

1) Proof of the first part of the lemma follows from the fact that, if encoded sequences of readings from all nodes are identical, they form one cluster.
2) The proof of the second part of the lemma requires two observations: a) The number of unique binary strings of length p is 2^p; and b) if $n \leq 2^p$, the maximum number of unique encoded sequences from n sensor-nodes is n. Thus, the maximum number of clusters is determined by $\min(n, 2^p)$.
3) Proof of the last part of the lemma is an immediate consequence of the *pigeon-hole principle*, where each unique binary string is a pigeon hole, and n data points are the n pigeons [40].

The lemma above shows that if we classify sensor nodes' readings from all sensors embedded in them, we may have too many groups. Now let us consider a more practical situation, where total number of readings from unreliable sensors of a given type is less than or equal to a predetermined number, say n_b. Under this situation, how many clusters are expected to form? Before we answer this question, let us introduce two notations that are counting facts. 平均情况下 有多少个cluster?

Fact 1: Let k and p be nonnegative integers such that $k \leq p$. The total number of selections, $c(k, p)$, from p unique objects such that no more than k objects are selected is given by

$$c(k, p) = \sum_{i=0}^{k} \binom{p}{i} \tag{6}$$

Fact 2: Let k and p be nonnegative integers such that $k \leq p$. The total number of objects selected in $tn(k, p)$ selections is

$$tn(k, p) = \sum_{i=0}^{k} i \cdot \binom{p}{i} \tag{7}$$

The following lemma answers the question raised earlier.

Lemma 2: At time instant t, let $Y^{(t)} = \{Y_1^{(t)}, Y_2^{(t)}, \cdots, Y_n^{(t)}\}$ be a set of n data-points of dimension p from a set of n sensor nodes $\{N_1, N_2, \cdots, N_n\}$. Also, assume that a sensor-type has no more than n_b readings from unreliable sensors.

Let k^* be an integer such that, $tn(k^*, p) \leq n_b \cdot p < tn(k^* + 1, p)$. If these p-dimension n data-points are clustered, the maximum number of clusters, $C(n, n_b, p)$, is given by,

$$C(n, n_b, p) = \min\left(c(k^*, p) + \omega(n, n_b, p), 2^p, n\right) \tag{8}$$

where $\omega(n, n_b, p) = \left\lfloor \frac{n_b \cdot p - tn(k^*, p)}{k^*+1} \right\rfloor$ ■

Before an outline of the proof is presented, let us consider an example.

Example 3: Let us consider the case of 10 sensor nodes with 4 sensors on each node. Also, assume that no more than 2 sensors of a type are unreliable. What is the maximum number of possible clusters?

We can see that $n_b = 2$. Thus, total number of unreliable sensors must be no more than $n_b \cdot p = 8$. In this case we find that $k^* = 1$, and $c(k^*, p) = c(1, p) = 5$, and $tn(k^*, p) = 4$ Thus, $\omega(n, n_b, p) = \left\lfloor \frac{n_b \cdot p - tn(k^*, p)}{k^*+1} \right\rfloor = \left\lfloor \frac{8-4}{2} \right\rfloor = 2$. Therefore, the maximum number of clusters is determined by the first term of the $\min(5 + 2, 16, 10) = 7$. However, it is interesting to note that the seven clusters are not unique. They can be selected in 3 different ways. ■

Proof: For maximizing the number of clusters, at first only one data-point is assigned at each cluster. To minimize the number of unreliable sensors, it is necessary to have a cluster with no unreliable sensor. Then, add one cluster at a time such that each cluster has only one bad data point. There are c$\binom{p}{1}$ clusters with only one bad sensor in each data-point. Next, select clusters with two bad sensors in each. This can be done in $\binom{p}{2}$ ways to obtain $\binom{p}{2}$ clusters. One can follow this cluster section process until we get a k^* such that $k^* = p$ or $tn(k*, p) \leq n_b \cdot p$. In the first case, we have achieved the maximum number of clusters.

In the second case, $tn(k^*, p) \leq n_b \cdot p < tn(k^* + 1, p)$, and we have used only $tn(k^*, p)$ sensors. Thus, another $(n_b \cdot p - tn(k^*, p))$ bad sensors may be added to maximize number of clusters. Since each additional cluster must have a group of $(k^* + 1)$ bad sensors, one can add $\left\lfloor \frac{n_b \cdot p - tn(k^*, p)}{k^*+1} \right\rfloor$ more clusters. This completes the outline of the proof. ■

To summarize, we have illustrated that when an aggregator receives p readings from a sensor node, the received p-dimensional data may form too many clusters even when readings are assigned binary values—reliable and unreliable. Thus, it may be hard to distinguish bad nodes and good nodes, which make data aggregation challenging.

In the rest of the section, we propose to partition the p-dimensional data into p sets of readings; then we use outlier detection algorithms presented in Section III-B on each set to partition them into two disjoint subsets: one from reliable sensors and the other from unreliable sensors.

B. Partitioning of Data by Sensor Type

We partition all the readings in Y for a sensor type i. For ease of visualization one can refer to Fig. 2, where ith horizontal slice is readings from all n nodes for sensor type i. *For ease of reference let these data be denoted by X_i, that is, $X_i =$* $[y_{i,1}, y_{i,2}, \cdots, y_{i,n}]$.

If one tries to cluster readings from only one sensor type, because of potentially many types of anomalies the data may form many clusters. Let us assume that X_i forms j_i clusters: $\{G_{i,1}, G_{i,2}, ..., G_{i,j_i}\}$. Now it is easy to see that p-types of sensors can form as many as $\prod_{i=1}^{p} j_i$ clusters. To simplify the situation we classify readings in each X_i into two classes, reliable and unreliable. To achieve this objective, we compute local outlier factor for each reading using LOF-based algorithms presented earlier.

C. Methods for Identification of Good and Bad Nodes

The major steps of our proposed methods for identification of good nodes and reliable sensors are outlined in Fig. 7. After receiving a batch of data Y, an aggregator partitions Y into p sets

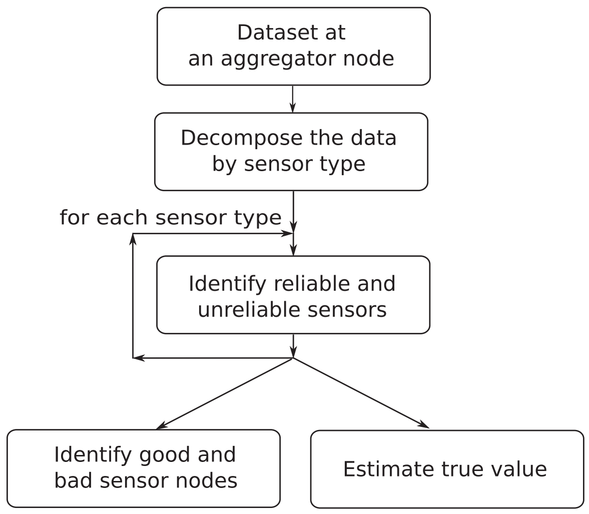

Dataset at an aggregator node
Decompose the data by sensor type
for each sensor type
Identify reliable and unreliable sensors
Identify good and bad sensor nodes
Estimate true value

Fig. 7. Proposed framework for identification of reliable and unreliable sensor nodes and true value estimation. The loop is executed a fixed number of times, once for each sensor-type.

$X_1, X_2, \cdots, X_p$, one for each sensor type i. Then the aggregator applies LOF algorithms (see Section III) on each data set X_i and computes a local outlier factor for each sensor (or each reading depending on the algorithm used). Next, using a dynamically determined threshold value, a sensor is designated reliable if the value of the local outlier factor is below the threshold value; otherwise, the sensor is designated unreliable. As shown in the Fig. 7, this step is repeated p times, once for each sensor type.

Once the sensors are labeled reliable and unreliable, the readings from unreliable sensors are designated outliers and form one set, say $G_{i,U}$. The remaining readings are from reliable sensors and denoted as $G_{i,R}$. Since $G_{i,U}$ and $G_{i,R}$ are two partitions of X_i, $G_{i,U} \bigcup G_{i,R} = X_i$ and $G_{i,U} \bigcap G_{i,R} = \emptyset$. For estimation of signal for sensor type i, we use readings in $G_{i,R}$.

For identification of good sensor nodes, readings in $G_{i,R}$ are associated with the sensor nodes $N_1, N_2, \cdots, N_n$. We create a set of nodes $N_{i,R}$ such that $N_j \in N_{i,R}$ if and only if $X_{i,j} \in G_{i,R}$. Similarly, we create a set of nodes $N_{i,U}$ such that $N_j \in N_{i,U}$ if and only if $X_{i,j} \in G_{i,U}$. Let N_R be the set of good nodes. The following lemma relates N_R with $N_{1,R}, N_{2,R}, \cdots, N_{p,R}$.

Lemma 3: For $1 \le i \le p$, if $N_{i,R}$ is the set of nodes with reliable sensors of type i, then the set of nodes with all reliable sensors is given by $N_R = \bigcap_{i=1}^{p} N_{i,R}$. ■

The proof of this lemma is quite intuitive. If a node N_j is in N_r, it is present in all the sets $N_{i,R}$, because N_R is constructed from their intersections. Thus, each sensor node N_i in the set N_R has p reliable sensors. On the other hand, if a node N_j is not in N_R, then there exists at least one i, such that N_j is not in $N_{i,R}$.

Similarly, the set of nodes with all bad sensor nodes is determined from $N_{i,U}$, for $1 \le i \le p$.

Lemma 4: For $1 \le i \le p$, if $N_{i,U}$ is the set of nodes with unreliable sensors of type i, then the set of nodes with all unreliable sensors is given by $N_U = \bigcap_{i=1}^{p} N_{i,U}$. ■

The proof of the above lemma is similar to the previous lemma and omitted.

Because Lemmas 3 and 4 are easily converted into algorithms to identify good nodes and bad nodes, we do not provide any description of the algorithms. The worst-case complexity of the algorithms arises when cardinalities of the sets $N_{i,U}$ (and $N_{i,R}$) are maximum, which is n. Since we are taking intersections of p such sets, the worst-case complexity of these algorithms is $O(np)$.

Nodes that are not in the either group are partially-good or partially-bad nodes. Recall that these partially-bad nodes have some reliable sensor(s) and some unreliable sensor(s).

1) Resistance on Total Number of Unreliable Sensors: The methods proposed in this section are robust against different collusion attacks as long as (i) the number of sensors affected by the attack is less than half of the total number of sensors of a type and (ii) readings from uncolluded sensors are relatively close to each other. This is true, because in Section III-A1 we noted that LOF algorithm can identify up to k outliers out of n data points, if $k < \lceil n/2 \rceil$. What is important to note is that we do not need to know the value of k a-priori; we can start with $k = \lfloor (n-1)/2 \rfloor$.

1.被攻击的节点个数小于1半;
2.未被攻击的节点的读数互相接近

In the next section, results from our empirical evaluation are presented. We have used $k = \lfloor (n-1)/2 \rfloor$ for our evaluations of the proposed methods, but the actual number of unreliable sensors was much less than $\lfloor (n-1)/2 \rfloor$. It was found that the proposed method is very effective in detecting good sensor-nodes, bad sensor-nodes, and partially-bad sensor nodes.

V. Empirical Evaluation of Good- and Bad-Node Identification Algorithms

First, we describe experimental settings common to all experiments in this section and for Section VI.

We perform two sets of simulation experiments. They use different sensor-error models and attack models. Their descriptions are provided in the beginning of Sections V-B and V-C, respectively. For each set, we report effectiveness of good and bad sensor-nodes identification method.[3]

A. Common Experimental Settings

We used Matlab 9.0 (R2015b) as our programing platform. In the sequel, $\mathcal{N}(\alpha, \beta^2)$ denotes a Gaussian distribution with bias α and standard deviation β. True signal is the temperature and dewpoint in Miami, Florida, USA on November 12, 2015 (see Fig. 8), which were obtained from the *Underground Weather* [41] website. Because, we have evaluated our methods with data colluded by simple as well as malicious attacks, we used synthetic datasets that were generated following procedures used in [9].

For the first set of the experiments, our noise and attack models for reliable and unreliable sensors are statistically identical to the noise and attack models used in [9]. Each simulation experiment is repeated 200 times and the results are averaged. Number of sensor nodes is $n = 20$. Number of readings from

[3]To the best of our knowledge, there is no prior published results for comparing with our results.

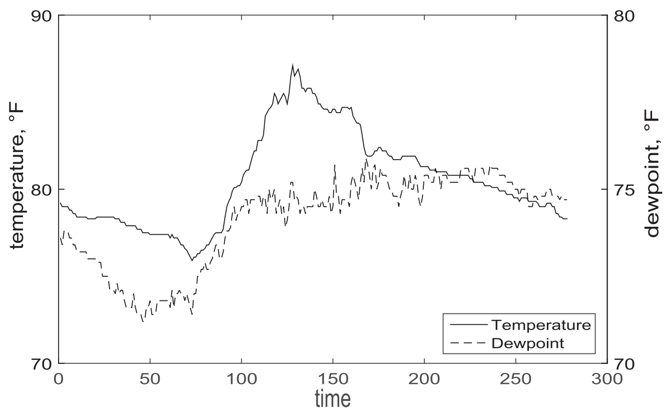


Fig. 8. Plot of 24-h temperature and dewpoint data from wunderground.com for Miami on November 12, 2015. Note that values on left- and right-sides are for temperature and dewpoint, respectively.

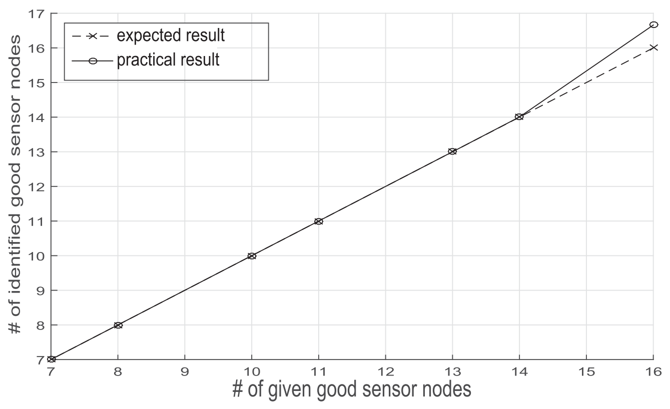


Fig. 9. Study I: Comparison of computed and actual good sensor nodes for collusion attack when $\sigma = 5$.

each sensor is $m = 288$, one reading every five minutes for 24 hours. Reported results are for 8 unreliable sensors.

The readings received by an aggregator is given by $x_s^{(t)} = r^{(t)} + e_s^{(t)}$, where $e_s^{(t)}$ is the error added to the true signal $r^{(t)}$. The distributions of $e_s^{(t)}$ for reliable and unreliable sensors are described in the corresponding sections.

To measure the efficacy of good and bad sensor-node identification, we report average of the number of sensor nodes that are correctly detected in 200 simulations. We present some of our typical observations from our extensive simulation results. For ease of comparison, we illustrate most of our results as line plots.

B. Experimental Settings and Observed Results for Study I

1) Signal and Noise Models: Below are noise models for reliable and unreliable sensors for this study.

1) Noise model for a reliable sensor at node N_s at time t is $e_s^{(t)} \sim \mathcal{N}(b_s, \sigma^2)$, where $b_s \sim \mathcal{N}(0, \sigma_b^2)$; noise for all reliable sensor(s) has same standard deviation.
2) Noise model for an unreliable sensor at node N_s at time t is $e_s^{(t)} \sim |\mathcal{N}(b'_s, 5 \cdot \sqrt{s} \cdot \sigma^2)|$, where $b'_s \sim \mathcal{N}(0, 3 \cdot \sigma_b^2)$; standard deviation of noise for each unreliable sensor is slightly different from other unreliable sensors.

In addition to identification of good and bad sensor-nodes, this signal and noise model is used for evaluation of data aggregation algorithms presented in Section VI.

2) Identification of Good and Bad Sensor Nodes: Recall that, all sensors in a good node are reliable, and all sensors in a bad node are unreliable. If a node is not good or bad, it is called a partially good sensor node. A partially good node has 1) two or more sensors, and 2) at least one reliable sensor and at least one unreliable sensor. In our experiments, the number of good sensor-nodes is varied from 7 to 16 (out of 20 nodes), and bad sensor-nodes is varied from 0 to 3. Table V shows seven different combinations of sensor nodes we used for the results reported here. Note that t-reliable means that temperature reading is reliable and d-reliable means that dewpoint reading is reliable. Number of reliable readings for each sensor type is varied from 12 to 18; in other words, the number of unreliable readings for each sensor-type is varied from 2 to 8.

As can be seen from Table V, for simple attack (where readings are altered to a higher or lower value) all good and bad nodes were identified correctly. For this results $\sigma = 5$, the worst case scenario in our experiments.

For malicious attack (where readings are altered to defy iterative filtering algorithms), good sensor nodes are identified correctly for all cases, except when number of good nodes is 16 (see plots in Fig. 9). In that case, average number of identified good nodes is 16.7. Our closer examination revealed that about 70% of the time outlier detection algorithm wrongly identifies one manipulated reading as reliable; to be more specific, the reading that is an average of 19 readings [9] is identified as reliable, because the value of the reading sent to the aggregator node being an average of other 19 readings, is quite close to the 18 reliable readings.

C. Experimental Settings and Observed Results for Study II

1) Signal and Noise Models: Our second set of experiments includes another set of noise models for both reliable and unreliable sensors. To be more specific,

1) Noise model for a reliable sensor at node N_s at time t is $e_s^{(t)} \sim \mathcal{N}(b_s, \sqrt{s} \cdot \sigma^2)$, where $b_s \sim \mathcal{N}(0, \sigma_b^2)$; standard deviation of noise for each reliable sensor is slightly different from others.
2) Noise model for an unreliable sensor at node N_s at time t is $e_s^{(t)} \sim |\mathcal{N}(b'_s, 5 \cdot \sigma^2)|$, where $b'_s \sim \mathcal{N}(0, 3 \cdot \sigma_b^2)$; noise for all unreliable sensor(s) has same standard deviation.

A closer look reveals that the noise for reliable sensors is much widely distributed than the noise for reliable sensors in the previous model. Thus, identification of good and bad sensor nodes as well as data aggregation expected to be more difficult. We created this difficult situation for testing robustness of our algorithms.

Similar to Study I, this signal and noise model is used for both good- and bad-node identification here and evaluation of data aggregation algorithms presented in Section VI.

2) Identification of Good and Bad Sensor Nodes: Observed detection-accuracy of good and bad sensor-nodes for these

TABLE V
STUDY I: COMPUTED AND ACTUAL GOOD, BAD, AND PARTIALLY GOOD SENSOR NODES FOR SIMPLE AND COLLUSION ATTACK WHEN $\sigma = 5$, THE WORST-CASE SCENARIO

Given combination of sensor nodes				Detected combination of sensor nodes							
				Simple attack				Collusion attack			
	partially good nodes				partially good nodes				partially good nodes		
good nodes	t-reliable	d-reliable	bad nodes	good nodes	t-reliable	d-reliable	bad nodes	good nodes	t-reliable	d-reliable	bad nodes
16	2	2	0	16	2	2	0	16.7	2.32	1.02	0.01
14	2	4	0	14	2	4	0	14	2	3.73	0.21
13	3	3	1	13	3	3	1	13	3	3.01	0.99
11	3	5	1	11	3	5	1	11	3	5.01	0.99
10	2	6	2	10	2	6	2	10	2	6.01	1.99
8	4	6	2	8	4	6	2	8	4	6	2
7	5	5	3	7	5	5	3	7	5	5	3

conditions is shown in Table VI and plotted in Fig. 10 for visual illustrations. The results suggest that for considered conditions the algorithm underestimates good sensor nodes while tends to overestimate the bad ones. This behavior can be explained by the nature of noise applied for this study. Recall, we introduced a coefficient $\sqrt{s}$ for the standard deviation for adding noise to readings from reliable sensors, which distorted readings of some of the reliable sensors and converted them to outliers. Thus, the number of *actual* outliers was higher than what is shown in Table VI. But the results were not too far off for the number of good nodes under 16.

VI. APPLICATION OF UNRELIABLE SENSOR DETECTION METHOD TO THE DATA AGGREGATION

As discussed in the introduction and in the review section, in the past data aggregation algorithms use iterative filtering to minimize influence of colluded data. In this section, we show that unreliable sensor detection method is very effective for data aggregation purpose.

We view, logically and functionally, data aggregation algorithms consist of two phases: 1) Detection and removal of readings provided by unreliable sensors, and 2) True value estimation with remaining data (see Fig. 11). Notations used for descriptions of the algorithms are shown in Table II.

A. *Data Aggregation Algorithms*

After the first phase of the algorithm is completed, reliable sensors in each node have been identified. In the last phase of the algorithm, a true value is estimated from readings of the reliable sensors for each time instance. Three outlier detection algorithms outlined in Section III-B, can be combined with IF, RDAM, and statistical estimation method to obtain a total of nine algorithms. Next we briefly outline them.

As in [9], to measure performance of the proposed data aggregation algorithms we use *Root Mean Squared error* (RMSE). We present some of our typical observations from our extensive simulation results. For ease of comparison we illustrate most of our results as line plots.

1) Detection of Unreliable Sensors and Iterative Filtering: The first phase of these algorithms uses one of the three outlier detection algorithms (DOSS, DOI, and DORV) predestined in Section IV-C to identify unreliable sensors. In the next phase, readings only from reliable sensors are used as inputs to an Iterative Filtering algorithm. In general, the estimated signals obtained from the three resulting algorithms have lower RMSE than the signals estimated by Iterative Filtering algorithm, if no readings are eliminated.

2) Detection of Unreliable Sensors and RDAM: Because RDAM is more robust than IF algorithms for reducing effect of malicious attacks, we have used it at the second phase of the data aggregation algorithm. We also use performance of RDAM as the benchmark for comparing performance of our algorithms. As earlier, one of the three outlier-detection algorithms (DOSS, DOI, and DORV) is used in the first phase. Thus, we have other three data aggregation algorithms. Extensive evaluations of these two-phase algorithms show that they estimate readings with much lower RMSE than the three algorithms obtained after combining outlier-detection algorithms with an Iterative Filtering algorithm.

3) Algorithm Based on Combination of Weights and Votes: This algorithm is obtained by combining the Row/Wise Votes method described in Section III-B3 and IF. In this combination, actually, there is no clear separation between the detection of outliers phase and the estimation of true values phase. Having a table of 'zero' or 'one' votes (see Table III) for each of the n sensors of type t, the algorithm uses its values as an indicator function for estimation of true signal (see Equation (9)).

$$\hat{r}^{(t)} = \sum_{s=1}^{n} w_s \cdot x_s^{(t)} \cdot votes(t) \tag{9}$$

If an indicator $votes(t)$ is 1 then the algorithm uses the corresponding reading, otherwise the reading is ignored because the sensor is considered unreliable.

4) Detection of Outliers and Statistical Estimation: To obtain this set of algorithms, we use one of the three outlier detection algorithms (DOSS, DOI, or DORV) in the first phase. Then remove readings provided by unreliable sensors, and use

TABLE VI
STUDY II: COMPUTED AND ACTUAL GOOD, BAD, AND PARTIALLY GOOD SENSOR NODES FOR SIMPLE AND COLLUSION ATTACK WHEN $\sigma = 5$, THE WORST-CASE SCENARIO

Given combination of sensor nodes				Detected combination of sensor nodes							
				Simple attack				Collusion attack			
	partially good nodes				partially good nodes				partially good nodes		
good nodes	t-reliable	d-reliable	bad nodes	good nodes	t-reliable	d-reliable	bad nodes	good nodes	t-reliable	d-reliable	bad nodes
16	2	2	0	13.29	3.03	3.07	0.62	13.03	2.95	3.24	0.79
14	2	4	0	12.13	3.08	4.15	0.66	13.18	2.91	3.30	0.62
13	3	3	1	11.79	3.32	3.50	1.40	12.59	3.47	3.44	0.51
11	3	5	1	10.03	3.32	5.14	1.53	10.17	3.38	5.86	0.60
10	2	6	2	9.10	2.34	6.05	2.52	9.12	2.39	6.96	1.54
8	4	6	2	7.48	4.00	5.83	2.70	7.47	4.11	6.05	2.38
7	5	5	3	6.63	4.80	4.79	3.79	6.50	5.02	5.08	3.41

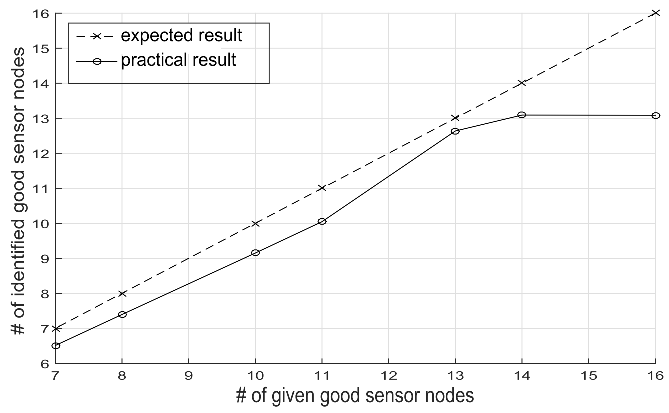


Fig. 10. Study II: Comparison of computed and actual good sensor nodes for collusion attack when $\sigma = 5$.

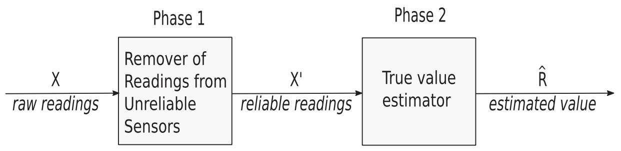


Fig. 11. Proposed two-phase data aggregation algorithms.

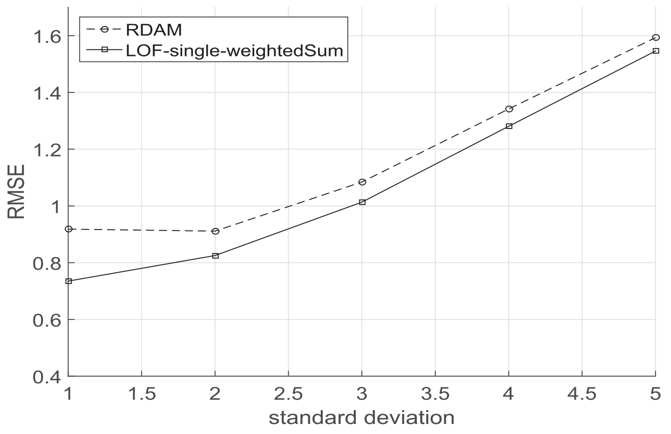


Fig. 12. Study I: Comparison of RMSE for RDAM and LOF-single-weighted sum algorithms for temperature data. Collusion attacks with eight unreliable sensors.

the remaining readings in the second phase, where statistical estimation method (described in Section II-C1) is used.

During our studies, best results were observed when we used DOSS in Section III-B1 with *Statistical Estimation* in Section II-C1. For reference, this algorithm is called *LOF-Single-weightedSum*.

B. Experimental Evaluation of Data Aggregation Algorithms

Next, we report performance of the true signal value estimation algorithm.

For iterative filtering algorithm and RDAM, we report results only for reciprocal discriminant function, $g(d) = d^{-1}$. (For details about different discriminant functions see [9]). The selection of this discriminant function is based on the fact that out of four discriminant functions used for evaluations in [9] this function had shown the best performance.

We extensively evaluated all the proposed data aggregation algorithms. Since the results are very close to each other, we report results only for *LOF-Single-weightedSum* algorithm (Section VI-A4) that produced slightly better performances consistently.

1) Results for Data From Study I (see Section V-B1): First, we conducted experiments using one-dimensional data, namely temperature data (illustrated in Fig. 8). For fair evaluations, our aggregation algorithm is compared with RDAM, currently available best sensor data aggregation algorithm [9]. Results reported are for 12 reliable sensors and 8 unreliable sensors and the attack model is malicious. As can be seen from plots in Fig. 12, our algorithm performs slightly better, but difference is insignificant; to be more specific, for our algorithm the RMSE values increased from 0.73 to 1.55 when standard deviation was increased from 1 to 5 and for the RDAM it varied from 0.91 to 1.6 for the same range of standard deviations.

When both sensor readings are evaluated simultaneously, the RMSE values are very similar and almost identical for temperature and dewpoint data set. At standard deviation 1 and 2, the RMSE value for dewpoint is higher than that of the temperature (see Fig. 13). We believe this is because of dewpoint values have a higher variation when they are compared to temperature values (see Fig. 8).

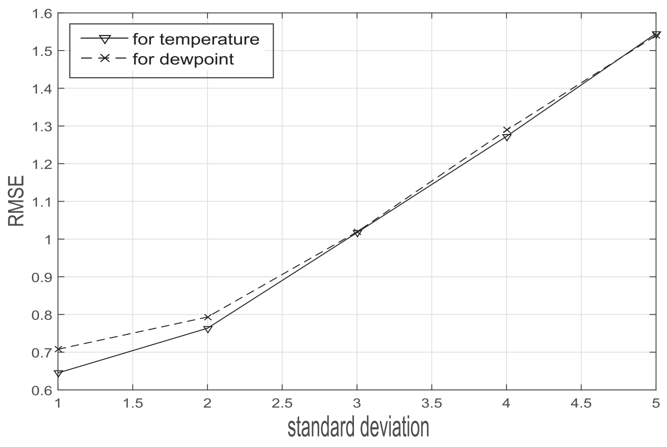


Fig. 13. Study I: RMSE for collusion attack for temperature and dewpoint for seven good and three bad sensor nodes.

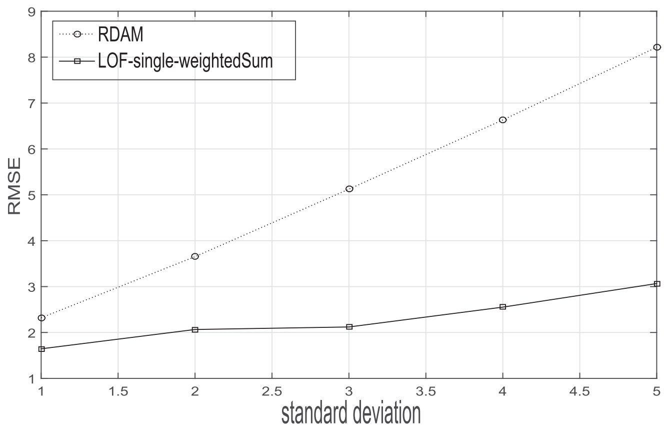


Fig. 14. Study II: Comparison of RMSE for RDAM and LOF-single-weighted sum algorithms for temperature data. Collusion attacks with eight unreliable sensors.

2) Results for Data From Study II (See Section V-C): As before, results reported are for 12 reliable sensors and 8 unreliable sensors and malicious attack model. First, we use only temperature dataset for comparing performance of our algorithms with that of the RDAM. As can be seen from plots in Fig. 14, our algorithm significantly outperforms RDAM. For our algorithm the RMSE values increases from about 1.8 to 3.0 when standard deviation is increased from 1 to 5 and for the RDAM they vary from about 2.2 to 8 for the same range of standard deviations. The reason for obtaining lower RMSE value for our algorithm is that we first eliminated outliers before estimating true signal. On the other hand, for the RDAM, too many readings from unreliable sensors adversely affected estimated true readings. Thus, the RMSE values are higher than the algorithms we have proposed.

VII. Discussion and Conclusion

A. Discussion

The outlier detection algorithms presented in Sections III-B and IV-C are for a cluster-head. We have discussed how a cluster-head can use them to aggregate data as well as to identify good, bad, and partially-bad nodes. Let us now consider a bigger picture, where these cluster-heads form a wireless sensor network.

The wireless sensor network could be composed of (i) homogeneous cluster-heads, that is, all cluster-heads are receiving data from identical[4] sensor-nodes, or (ii) heterogeneous cluster-heads, that is, different cluster-heads are receiving data from different sensor nodes. An interesting question is: can these cluster-heads cooperate to form an intrusion detection system? We believe the answer is yes.

Two possible types of sensor networks–(a) distributed networks and (b) centralized networks–can be formed for intrusion detection and monitoring purpose. Next we give a brief outline for possible intrusion detection methods for each type of networks. We assume that each sensor node has a *unique identity* (UID) number, and each cluster-head has a list of UIDs of sensor nodes that are sending data to it. Also, a cluster-head maintains a list that has UIDs of sensor nodes and their sensors that appear to be unreliable. Let this list be called the U-list of the cluster-head.

In a centralized network-intrusion detection-system, the monitoring would be done by a central node, which may not be a cluster-head. If a cluster-head observes any difference between its current U-list and the previous U-list, it (directly or indirectly) communicates observed difference to the central monitoring node, and the central monitoring node utilizes received observations from all cluster-heads of the network for intrusion detection. We plan to develop such algorithms in the future.

The biggest problem with a centralized system is scalability. A possible solution would be using a distributed algorithm where each cluster-head is connected to a subset of cluster-heads. Each cluster-head maintains its U-list as before, but also it maintains an *intrusion factor* (IF) that it computes from the data it receives from its neighbors. Each cluster-head exchanges with its neighbors the difference between current and previous U-lists as well as current IF. Then all cluster-heads recalculate their IF for using in the next iteration. Assuming that the sensor network is connected, local IFs will propagate over the network, and all cluster-heads will reach a consensus about an intrusion event.

B. Conclusion

Most IoE devices have multiple types of inexpensive sensors embedded in them, which are not very durable and vulnerable to attacks, including malicious attacks. To overcome unreliability of sensors, readings are gathered from multiple sensors and a true value is estimated by an aggregator. Most recent aggregation methods include iterative filtering algorithms and robust data aggregation methods (see [9], and the references therein). However, obtained estimation of any method that does not explicitly removes readings from unreliable sensors, could be further from the true signal.

[4]All the sensor nodes were manufactured with same set of sensors, but may have some minor insignificant variations.

We use a modified version of a recently developed local outlier factor computation algorithm to identify outliers among the readings. We designate a sensor unreliable, if its readings are outliers. We have proposed several algorithms for selecting unreliable sensors. Once reliable sensors are identified, we use their readings for true signal value estimation.

We consider a sensor node is *good*, if all of its sensors are reliable. We have developed a method to identify good sensor nodes. The algorithm is useful for identification of nodes that have all unreliable sensors.

We have empirically evaluated the proposed algorithm for identification of good- and bad-sensor nodes and have observed that it is very effective. In one study, our data aggregation method has significantly out-performed best aggregation algorithm (RDAM [9]). Our data aggregation method is very effective for overcoming malicious attacks, because it eliminates readings from anomalous (unreliable or compromised) sensors.

In the future, we want to investigate proposed good and bad sensor-node identification method for application to security and surveillance. We expect that as the sensor nodes are indirectly and remotely monitored, their change in reliability can be associated to security threats and thus, a tool for surveillance of the network or the environment where the sensors are physically located. An implementation of this method in our own sensor network is in progress.

Acknowledgment

The authors are grateful to Prof. Sanjay Jha and Dr. Mohsen Rezvani for providing them their Matlab programs, which were used for the simulation results in [9]. Also, they are grateful to the reviewers and the Associate Editor for their constructive comments that have significantly improved the presentation and contents of the paper.

References

[1] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–120, Apr. 2014.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, Fourth Quarter 2015.

[3] S. W. Abeyruwan, D. Sarkar, F. Sikder, and U. Visser, "Semi-automatic extraction of training examples from sensor readings for fall detection and posture monitoring," *IEEE Sens. J.*, vol. 16, no. 13, pp. 5406–5415, Jul. 2016.

[4] S. Abeyruwan, F. Sikder, U. Visser, and D. Sarkar, "Activity monitoring and prediction for humans and NAO humanoid robots using wearable sensors," in *Proc. 28th Int. FLAIRS Conf.*, 2015, pp. 342–347.

[5] F. Sikder and D. Sarkar, "Log-sum distance measures and its application to human-activity monitoring and recognition using data from motion sensors," *IEEE Sens. J.*, vol. 17, no. 14, pp. 4520–4533, Jul. 2017.

[6] M. Z. A. Bhuiyan and J. Wu, "Collusion attack detection in networked systems," in *Proc. IEEE 14th Int. Conf. Dependable, Auton. Secure Comput.*, 2016, pp. 286–293.

[7] F. Khedim, N. Labraoui, and M. Lehsaini, "Dishonest recommendation attacks in wireless sensor networks: A survey," in *Proc. 12th Int. Symp. Program. Syst.*, Apr. 2015, pp. 1–10.

[8] A. Mahapatro and P. M. Khilar, "Fault diagnosis in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2000–2026, Fourth Quarter 2013.

[9] M. Rezvani, A. Ignatovich, E. Bertino, and S. Jha, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 98–110, Jan./Feb. 2015.

[10] K. Ni *et al.*, "Sensor network data fault types," *ACM Trans. Sens. Netw.*, vol. 5, no. 3, 2009, Art. no. 25.

[11] A. Yessembayev, D. Sarkar, and F. Sikder, "Detection of unreliable sensors and sensor-nodes in presence of malicious attacks, and its application to data aggregation," Univ. Miami, Coral Gables, FL, USA, Tech. Rep., 2016.

[12] D. Wagner, "Resilient aggregation in sensor networks," in *Proc. 2nd ACM Workshop Security Ad Hoc Sens. Netw.*, 2004, pp. 78–87.

[13] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Trans. Sensor Netw.*, vol. 6, no. 3, 2010, Art. no. 23.

[14] X. R. Wang, J. T. Lizier, O. Obst, M. Prokopenko, and P. Wang, "Spatiotemporal anomaly detection in gas monitoring sensor networks," in *Proc. 5th Eur. Conf. Wireless Sens. Netw.*, 2008, pp. 90–105.

[15] R. Jurdak, X. Wang, O. Obst, and P. Valencia, "Wireless sensor network anomalies: Diagnosis and detection strategies," *Intell.-Based Syst. Eng.*, vol. 10, pp. 309–325, 2011.

[16] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 51–59.

[17] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Localized outlying and boundary data detection in sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1145–1157, Aug. 2007.

[18] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 159–170, Second Quarter 2010.

[19] Y. Zhang, N. A. S. Hamm, N. Meratnia, A. Stein, M. van de Voort, and P. J. M. Havinga, "Statistics-based outlier detection for wireless sensor networks," *Int. J. Geograph. Inf. Sci.*, vol. 26, no. 8, pp. 1373–1392, 2012.

[20] A. Yessembayev, "Secure data aggregation algorithms for sensor networks in the presence of collusion attacks using local outlier factor," master's thesis, Dept. Comput. Sci., Univ. Miami, Coral Gables, FL, USA, Dec. 2015.

[21] A. Yessembayev and D. Sarkar, "Secure data aggregation algorithms for sensor networks in the presence of collusion attacks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Mar. 2016, pp. 1–6.

[22] K.-F. Ssu, C.-H. Chou, H. C. Jiau, and W.-T. Hu, "Detection and diagnosis of data inconsistency failures in wireless sensor networks," *Comput. Netw.*, vol. 50, no. 9, pp. 1247–1260, 2006.

[23] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1132–1144, Aug. 2010.

[24] R. Isermann, "Model-based fault-detection and diagnosis—Status and applications," *Annu. Rev. Control*, vol. 29, no. 1, pp. 71–85, 2005.

[25] M. Panda and P. M. Khilar, "Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test," *Ad Hoc Netw.*, vol. 25, pp. 170–184, 2015.

[26] S. Ji, S.-F. Yuan, T.-H. Ma, and C. Tan, "Distributed fault detection for wireless sensor based on weighted average," in *Proc. 2nd Int. Conf. Netw. Security Wireless Commun. Trusted Comput.*, 2010, vol. 1, pp. 57–60.

[27] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Proc. Workshop Dependability Issues Wireless Ad hoc Netw. Sens. Netw.*, 2006, pp. 65–72.

[28] W. Li, F. Bassi, D. Dardari, M. Kieffer, and G. Pasolini, "Defective sensor identification for WSNs involving generic local outlier detection tests," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 1, pp. 29–48, Mar. 2016.

[29] M.-H. Lee and Y.-H. Choi, "Fault detection of wireless sensor networks," *Comput. Commun.*, vol. 31, no. 14, pp. 3469–3475, 2008.

[30] A. De Paola, S. Gaglio, G. L. Re, F. Milazzo, and M. Ortolani, "Adaptive distributed outlier detection for WSNs," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 902–913, May 2015.

[31] T. Tošić, N. Thomos, and P. Frossard, "Distributed sensor failure detection in sensor networks," *Signal Process.*, vol. 93, no. 2, pp. 399–410, 2013.

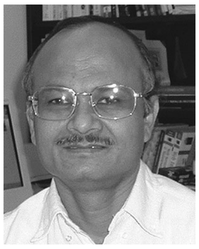
[32] E. Ayday and F. Fekri, "Iterative trust and reputation management using belief propagation," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 3, pp. 375–386, May/Jun. 2012.

[33] C. de Kerchove and P. Van Dooren, "Iterative filtering in reputation systems," *SIAM. J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 1812–1834, 2010.

[34] R.-H. Li, J. X. Yu, X. Huang, and H. Cheng, "Robust reputation-based ranking on bipartite rating networks," in *Proc. SIAM Int. Conf. Data Mining*, 2012, vol. 12, pp. 612–623.

[35] V. Barnett and T. Lewis, *Outliers in Statistical Data*. New York, NY, USA: Wiley, 1994.
[36] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 93–104.
[37] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.
[38] J. A. Hartigan, *Clustering Algorithms*, 99th ed. New York, NY, USA: Wiley, 1975.
[39] D. Hawkins, *Identification of Outliers*. London, U.K.: Chapman & Hall, 1980.
[40] K. P. Bogart, *Introduction to Combinatorics*. Orlando, FL, USA: Harcourt Brace Jovanovich, 1990.
[41] The Weather Channel, Miami, FL, USA, 2015. [Online]. Available: www.wunderground.com/us/fl/miami

Anes Yessembayev received the Specialist degree in computer science from Karaganda State Technical University, Karaganda, Kazakhstan, in May 1998, and the M.S. degree in computer science from the University of Miami, Coral Gables, FL, USA, in December 2015. He is currently working for the government of Kazakhstan. His research interests include sensors networks and their applications. He is especially interested in secure data aggregation algorithms for sensor networks.



Dilip Sarkar (SM'96) received the B.Tech. (Hons.) degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in May 1983, the M.S. degree in computer science from the Indian Institute of Science, Bangalore, India, in December 1984, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA, in May 1988. From January 1985 to August 1986, he was a Ph.D. student with Washington State University, Pullman, WA, USA. He is currently an Associate Professor of computer science with the University of Miami, Coral Gables, FL, USA. He has served on the program committees of the IEEE International Conference on Communications, the IEEE Globecom, the IEEE International Conference on Multimedia and Expo, the International Conference on Computer Communications and Networks, and the IEEE INFOCOM for many years. His research interests include concurrent transport protocols, parallel and distributed processing, neural networks, wireless sensors networks and their applications, and security of virtual machines. In these areas, he has guided several theses and has authored numerous papers. He served as a Guest Editor of a special issue of the *Computer Communications* on Concurrent Multipath Transport.

Faisal Sikder received the B.Sc. (Hons.) and M.S. degrees in computer science and engineering from the University of Dhaka, Dhaka, Bangladesh, in 2008 and 2010, respectively, and the M.S. and Ph.D. degrees in computer science from the University of Miami, Coral Gables, FL, USA, in 2014 and 2017, respectively. He is currently a Senior R&D Software Engineer with 3Z Telecom, Miramar, FL, USA. His research interests include machine learning, wearable sensors and their applications, the Internet of things (IoT), and distributed computing in the IoT.